Integrating control design and real-time computing : a robust control approach

Daniel Simon, Olivier Sename, David Robert, Mohamed El Mongi Ben Gaïd

Integrating control design and real-time computing : a robust control approach

Daniel Simon * **Olivier Sename** ** **David Robert** **
Mongi Ben Gaid *

* *INRIA Rhône-Alpes, NeCS team, Inovallée, Montbonnot, 38334 Saint-Ismier Cedex, France*
e-mail : Daniel.Simon@inrialpes.fr

** *GIPSAlab - Department of Control Systems, ENSIEG, BP 46 38402 Saint Martin d'Heres Cedex, France*
e-mail : olivier.sename@inpg.fr

Abstract: Control systems running on a computer are subject to timing disturbances coming from implementation constraints. Fortunately closed-loop systems behave robustly w.r.t. modelling errors and disturbances, and the controller design can be performed to explicitly enhance robustness against specific uncertainties. On one hand robustness in process controllers can be used to comply with weakly modelled timing uncertainties. On the other hand, the principle of robust closed-loop control can also be applied to the real-time scheduler to provide on-line adaption of some scheduling parameters, with the objective of controlling the computing resource allocation. As varying sampling appears to be a key actuator to control the computing resource, a varying sampling control design based on LPV gain scheduling design is provided. The feasibility of the approach is assessed through several examples using simulation and real experiments.

Keywords: control/computing co-design, robust control, sampling varying, real-time

1. INTRODUCTION

Digital control systems can be implemented as a set of tasks running on top of an off-the-shelf real-time operating system (RTOS) using fixed-priority and preemption. The performance of the control, e.g measured by the tracking error, and even more importantly its stability, strongly relies on the values of the sampling rates and sensor-to-actuator latencies (the latency we consider for control purpose is the delay between the instant when a measure q_n is taken on a sensor and the instant when the control signal $U(q_n)$ is received by the actuators Åström and Wittenmark [1997]). Therefore, it is essential that the implementation of the controller respects an adequate temporal behaviour to meet the expected performance. However implementation constraints such as multi-rate sampling, preemption, synchronisation and various sources of delays make the run-time behaviour of the controller very difficult to accurately predict. However as we deal with closed-loop controllers we may take advantage of the *robustness* of such systems to design and implement flexible and adaptive real-time control architectures.

This paper deals with some robust and adaptive solutions for real-time scheduling and control co-design. In the next sections we review some properties of closed-loop controllers in contrast with real-time implementation constraints. Some recent results in control and scheduling co-design are recalled in section 3. Section 4 gives an overview of a feedback scheduling strategy aimed to on-line adapt the tasks period according to the computing resource activity. This approach is then applied in the design of a robot controller in section 4.4, for which an experimental feedback scheduler implementation inside a hardware in the loop real-time simulator is experimented. As variable sampling appears to be a decisive actuator in CPU load control, a robust design of variable rate control based on the LPV approach is described in section 5. Is is assessed via an real experiment using an inverted pendulum in section 5.3. Future research directions are sketched to conclude the paper.

2. CONTROL AND COMPUTING CONSTRAINTS

Closed-loop digital control systems use a computer to periodically sample sensors, compute a control law and send control signals to the actuators of a continuous time physical process. The control algorithm can be either designed in continuous time and then discretized or directly synthesised in discrete time taking account of a model of the plant sampled by a zero-order holder. Control theory for linear systems sampled at fixed rates has been established a long time ago, e.g. Åström and Wittenmark [1997].

Assigning an adequate value for the sampling rate is a decisive duty as this value has a direct impact on the control performance and stability. While an absolute

⋆ This work is partially supported by the Safe_NeCS project funded by the ANR under grant ANR-05-SSIA-0015-03

lower limit for the sampling rate is given by Shannon's theorem, in practise rules of thumb are used to give a useful range of control frequencies according to the process dynamics and to the desired closed-loop bandwidth (see for example section 4.4.2). A commonly shared idea is that lower are the control period and latencies, better is the control performance (e.g. measured by the tracking error or disturbances rejection [1]).

2.1 Digital Control of Continuous Systems

To implement a controller, the basic idea consists in running the whole set of control equations in a unique periodic real-time task whose clock gives the controller sampling rate. In fact, all parts of the control algorithm do not have an equal weight and urgency w.r.t. the control performance. To minimise the latency, a control law can be basically implemented as two real-time blocks, the urgent one sends the control signal directly computed from the sampled measures, while updating the state estimation or parameters can be delayed or even more computed less frequently Åström and Wittenmark [1997].

In fact, a complex system involves sub-systems with different dynamics which must be further coordinated Törngren [1998]. Assigning different periods and priorities to different blocks according to their relative weight allows for a better control of critical latencies and for a more efficient use of the computing resource Simon et al. [1998]. However in such cases finding adequate periods for each block is out of the scope of current control theory and must be done through case studies, simulation and experiments.

Latencies have several sources: the first one comes from the computation duration itself, and worst case execution times are difficult to get. In multi-tasking systems they come from preemption due to concurrent tasks with higher priority, from precedence constraints and from synchronisation. Another source of delays is the communication medium and protocols when the control system is distributed on a network of connected devices. In particular it has been observed that in synchronous multi-rate systems the value of sampling-induced delays show complex patterns and can be surprisingly long Wittenmark [2001].

2.2 Control and Timing Uncertainty

While timing uncertainties have an impact on the control performance they are difficult to be accurately modelled or constrained to lie inside precisely known bounds. Thus it is worth examining the sensitivity of control systems w.r.t. timing fluctuations.

Control systems are often cited as examples of "hard real-time systems" where jitter and deadline violations are strictly forbidden. In fact experiments show that this assumption may be false for closed-loop control. Any practical feedback system is designed to obtain some stability margin and robustness w.r.t. the plant parameters uncertainty. This also provides robustness w.r.t. timing uncertainties: closed-loop systems are able to tolerate some amount of sampling period and computing delays deviations, jitter and occasional data loss with no loss of stability or integrity, e.g. Cervin [2003]: their behaviour can still be considered as correct as long as the sample-induced disturbances stay inside the performance specification bounds.

Therefore the hard real-time assumption must be softened to better cope with the reality of closed-loop control. For example they can be changed for "weakly hard" constraints: absolute deadlines are replaced by statistical ones, e.g. the allowable output jitter compliant with the desired control performance or the number of allowed deadlines miss over a specified time window Bernat et al. [2001]. Note that to be fully exploited, weakly hard constraints should be associated with a decisional process: tasks missing their deadline can be for example delayed, aborted or skipped according to their impact on the control law behaviour, e.g. as analysed in Cervin [2005].

Finding the values of such weakly hard constraints for a given control law is currently out of the scope of current control theory in the general case. However the intrinsic robustness of closed-loop controllers allows for complying with softened timing constraints specification and flexible scheduling design.

2.3 Control and Scheduling

From the implementation point of view, real-time systems are usually modelled by a set of periodic tasks assigned to one or several processors and a worst case response times technique is used to analyse fixed-priority real-time systems. Well known scheduling policies, such as Rate Monotonic for fixed priorities and EDF for dynamic priorities, assign priorities according to timing parameters, respectively sampling periods and deadlines. They are said to be "optimal" as they maximise the number of tasks sets which can be scheduled with respect of deadlines, under some restrictive assumptions. Unfortunately they are not optimised for control purpose.

They hardly take into account precedence and synchronisation constraints which naturally appear in a control algorithm. The relative urgency or criticality of the control tasks can be unrelated with the timing parameters. Thus, the timing requirements of control systems w.r.t. the performance specification do not fit well with scheduling policies purely based on schedulability tests. It has been shown through experiments, e.g. Cervin [2003], that a blind use of such traditional scheduling policy can lead to an inefficient controller implementation; on the other hand a scheduling policy based on application's requirements, associated with a right partition of the control algorithm into real-time modules may give better results. It is often the case that improving some computing related features is in contradiction with another one targeted to improve the control behaviour. For example the case studies examined in Buttazzo and Cervin [2007] show that an effective method to minimise the output control jitter consists in systematically delaying the output delivery at the end of the control period : however this method also introduces a systematic one period input/output latency and therefore most often provides the worst possible control performance among the set of studied strategies.

[1] This assumption can be enforced by providing a suitable control parameters tuning, as discussed at the end of the paper

Another example of unsuitability between computing and control requirements arises when using priority inheritance or priority ceiling protocols to bypass priority inversion due to mutual exclusion, e.g. to ensure the integrity of shared data. While they are designed to avoid dead-locks and minimise priority inversion lengths, such protocols jeopardise at run-time the initial schedule which was carefully designed to meet control requirements. As a consequence latencies along some control paths can be largely increased leading to a poor control performance or even instability.

Finally off-line schedulability analysis rely on a right estimation of the tasks worst case execution time. Even in embedded systems the processors use caches and pipelines to improve the average computing speed while decreasing the timing predictability. Another source of uncertainty may come from some pieces of the control algorithm. For example, the duration of a vision process highly depends on incoming data from a dynamic scene. Also some algorithms are iterative with a badly known convergence rate, so that the time before reaching a predefined threshold is unknown (and must be bounded by a timeout). In a dynamic environment, some control activities can be suspended or resumed and control algorithms with different costs can be scheduled according to various control modes leading to large variations in the computing load.

Thus real-time control design based on worst case execution time, maximum expected delay and strict deadlines inevitably leads to a low average usage of the computing resource and to a poor adaptivity w.r.t. a complex execution environment. All these drawbacks call for a better integration of control goals and computing capabilities through a co-design approach.

3. CONTROL/SCHEDULING CO-DESIGN

3.1 Scheduling Parameters Assignment

This mainly concerns the integration of control performance knowledge in the scheduling parameters assignment. Indeed, once a control algorithm has been designed, a first job consists in assigning timing parameters, i.e. periods of tasks and deadlines, so that the controller's implementation satisfies the control objective. This may be done off-line or on-line.

In off-line control/scheduling co-design setting adequate values for the timing parameters rapidly falls into case studies based on simulation and experiments. For instance in Ryu et al. [1997] off-line iterative optimisation is used to compute an adequate setting of periods, latencies and gains resulting in a requested control performance according to the available computing resource and implementation constraints. Also in Sandström and Norström [2002] the temporal requirements of the control system are described using complex temporal attributes (e.g. nominal period and allowed variations, precedence constraints. . .): this model is then used by an off-line iterative heuristic procedure to assign the scheduling parameters (e.g. priorities and offsets) to meet the constraints.

Concerning co-design for on-line implementation, recent results deal with varying sampling rates in control loops in the framework of linear systems: for example Schinkel et al. [2002] show that, while switching between two stable controllers, too frequent control period switches may lead to unstability. Unfortunately most real-life systems are non-linear and the extrapolation of timing assignment through linearisation often gives rough estimations of allowable periods and latencies or even can be meaningless. In fact, as shown later in the examples, the knowledge of the plant's behaviour is necessary to get an efficient control/scheduling co-design.

3.2 Feedback Scheduling

Besides traditional assignment of fixed scheduling parameters, more flexible scheduling policies have been investigated. Let us cite e.g. Buttazzo and Abeni [2000] where the elasticity of the tasks' periods enables for controlling the quality of service of the system as a function of the current estimated load. While such an approach is still working in open loop w.r.t. a controlled plant, the on-line combination the control performance and implementation constraints lead to the feedback scheduling approach.

This approach has been initiated both from the real-time computing side Lu et al. [2002] and from the control side Cervin and Eker [2000], Eker et al. [2000], Cervin et al. [2002]. The idea consists in adding to the process controller an outer sampled feedback loop ("scheduling regulator") to control the scheduling parameters as a function of a QoC (Quality of Control) measure. It is expected that an on line adaption of the scheduling parameters of the controller may increase its overall efficiency w.r.t. timing uncertainties coming from the unknown controlled environment. Also we know from control theory that closing the loop may increase performance and robustness against disturbances when properly designed and tuned (otherwise it may lead to instability).

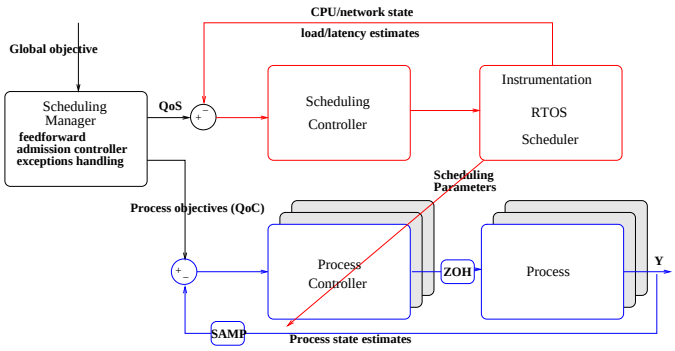

Fig. 1. Hierarchical control structure

Figure 1 gives an overview of a feed-back scheduler architecture where an outer loop (the *scheduling controller*) adapts in real-time the scheduling parameters from measurements taken on the computer's activity, e.g. the computing load [2]. Besides this controller working periodically (at a rate larger than the sampling periods of the plant control tasks), the system's structure may evolve along a discrete time scale upon occurrence of events, e.g. for new tasks admission or exception handling. These decisional processes may be handled by another real-time task, the

[2] Ideally it would be also fed by measures related to the quality of control, thus really providing integrated control and scheduling

scheduling manager, which is not further detailed in this paper. Notice that such a manager may give a reference to the controller resource utilisation.

The design problem can be stated as control performance optimisation under constraint of available computing resources. Early results come from Eker et al. [2000] where a problem of optimal control under computation load constraints is theoretically solved by a feedback scheduler, but leads to a solution too complex to be implemented in real-time. Then Cervin [2003] shows that this optimal control problem can be often simply implemented by computing the new tasks periods by the rescaling:

$$h_i^{k+1} = h_i^k \frac{U}{U_{sp}}$$

where U_{sp} is the utilisation set-point and U the estimated CPU load. The feedback scheduler then controls the processor utilisation by assigning task periods that optimise the overall control performance. This approach is well suited for a "quasi-continuous" variation of the sampling periods of real-time tasks under control of a preemptive real-time operating system.

Another approach has been used in the framework of the so-called *(m,k)-firm* schedulability policy, where the scheduling strategy ensures the successful execution of at least m instances of a given task (or message sending) for each time window of length k slots. Hence a selective data drop policy (as in Jia et al. [2007]) or a computing power allocation to selected tasks (as in Ben Gaid et al. [2006]) can be used to perform optimal control of a plant under constraint of computing or communication limitations. This latter approach is well suited for non-preemptive scheduling of control tasks and for networked control systems subject to messages loss : the tasks or messages are scheduled to jointly perform congestion avoidance and optimal control.

Indeed in all cases the adaptive behaviour of a feedback scheduler, associated with the relative tolerance of the control system w.r.t. the implementation induced timing uncertainties, allows for the design and implementation of real-time control systems based on their average execution behaviour rather than on pessimistic worst cases estimates.

4. ROBUST CONTROL OF THE COMPUTING RESOURCE

Feedback scheduling is a dynamic approach allowing a better using the computing resources, in particular when the workload changes e.g. due to the activation of an admitted new task. Indeed, the CPU activity will be controlled according to the resource availability by adjusting scheduling parameters (i.e. period) of the plant control tasks.

In the approach here proposed, a way to take into account the resource sharing over a multitasking process is developed. In what follows, the control design issue is described including the control structure, the specification of control inputs and measured outputs, as well as the modelling step.

4.1 Control Structure

In Fig 2 scheduling is viewed as a dynamical system between control task frequencies and processor utilisation. As far as the adaptation of the control tasks is concerned, the load of the other tasks is seen as an output disturbance.

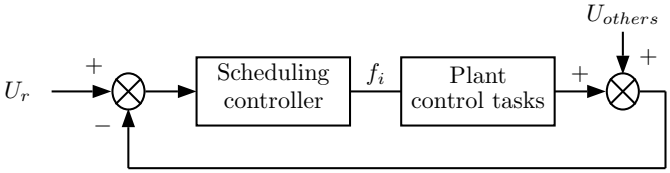

Fig. 2. Feedback scheduling bloc diagram

4.2 Sensors and Actuators

As stated in section 2.3, priorities must be assigned to control tasks according to their relative urgency ; this ordering remains the same in the case of a dynamic scheduler. Dynamic priorities, e.g. as used in EDF, only alter the interleaving of running tasks and will fail in adjusting the computing load w.r.t. the control requirements. In consequence we have elected the tasks periods to be the main actuators of the system running on top of a fixed priority scheduler [3].

As the aim is to adjust on-line the sampling periods of the controllers in order to meet the computing resource requirements, the control inputs are thus the periods of the control tasks.
The measured output is the CPU utilisation. Let us first recall that the scheduling is here limited to periodic tasks. In this case the processor load induced by a task is defined by $U = \frac{c}{h}$ where c and h are the execution time and period of the task. Hence processor load induced by a task is estimated, in a similar to way Cervin et al. [2002], for each period h_s of the scheduling controller, as:

$$\hat{U}_{kh_s} = \lambda\, \hat{U}_{(k-1)h_s} + (1-\lambda)\, \frac{\overline{c}_{kh_s}}{h_{(k-1)h_s}} \qquad (1)$$

where h is the sampling frequency currently assigned to the plant control task (i.e. at each sampling instant kh_s) and $\overline{c}$ is the mean of its measured job execution-time. λ is a forgetting factor used to smooth the measure.

4.3 Control Design and Implementation

The proposed control design method for feedback scheduling is here developed. First one should note that, as shown in Simon et al. [2003], if the execution times are constant, then the relation, $U = \sum_{i=1}^{n} C_i f_i$ (where $f_i = 1/h_i$ is the frequency of the task) is a linear function (while it would not be the case if expressed as a function of the task

[3] Possible secondary actuators are variants of the control algorithms, with different QoS contributions to the whole system. Such variants should be handled by the scheduling manager working on a discrete events time scale

periods). Therefore, using (1), the estimated CPU load is given as:

$$\hat{U}(kh_S) = \frac{(1-\lambda)}{z-\lambda} \sum_{i=1}^{n} \overline{c}_i(kh_S) f_i(kh_S) \tag{2}$$

An illustration, for the case of a single control task system, is given in figure 3 where the estimated execution-times are used on-line to adapt the gain of the controller for the original CPU system (2) (this allows to compensate the variations of the job execution time).

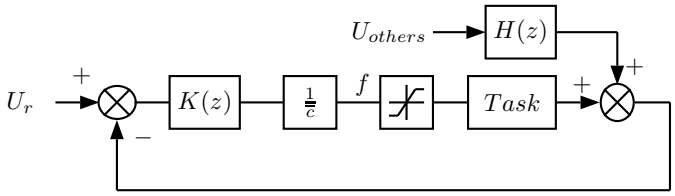

Fig. 3. Control scheme for CPU resources

As $\overline{c}$ depends on the run-time environment (e.g. processor speed) a "normalised" linear model of the task i (i.e independent on the execution time), G_i, is used for the scheduling controller synthesis where $\overline{c}$ is omitted and will be compensated by on-line gain-scheduling ($1/\overline{c}$) as shown below.

$$G_i(z) = \frac{\hat{U}(z)}{f_i(z)} = \frac{1-\lambda}{z-\lambda}, \quad i = 1, \ldots, n \tag{3}$$

According to this control scheme, the design of the controller K can be made using any advanced control methodology. For the considered application, we have chosen the well known H_∞ control theory which can lead to a robust controller w.r.t modelling errors (see Zhou et al. [1996] for details on H_∞ control). Moreover it provides good properties in presence of external disturbance, as it is emphasised in the example below.

4.4 Feedback Scheduling a Robot Controller

We consider here a seven degrees of freedom Mitsubishi PA10 robot arm that has been previously modelled and calibrated Simon et al. [2005].

Plant Modelling and Control Structure The problem under consideration is to track a desired trajectory for the position of the end-effector. Using the Lagrange formalism the following model can be obtained:

$$\Gamma = M(q)\ddot{q} + Gra(q) + C(q, \dot{q}) \tag{4}$$

where q stands for the positions of the joints, M is the inertia matrix, Gra is the gravity forces vector and C gathers Coriolis, centrifugal and friction forces.

The structure of the (ideal) linearising controller includes a compensation of the gravity, Coriolis/centrifugal effect and Inertia variations as well as a Proportional-Derivative (PD) controller for the tracking and stabilisation problem, of the form:

$$\Gamma = Gra(q) + C(q, \dot{q}) + K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}), \tag{5}$$

leading to the linear closed-loop system $M(q)\ddot{q} = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q})$.

This controller is divided in four tasks, i.e. a specific task is considered for the PD control, for the gravity, Inertia and Coriolis compensations, in order to use a multi-rate controller. In this first cautious feedback scheduling scheme, only the periods of the compensation tasks will be adapted, as they are time consuming compared with the PD task while being less critical for the stability.

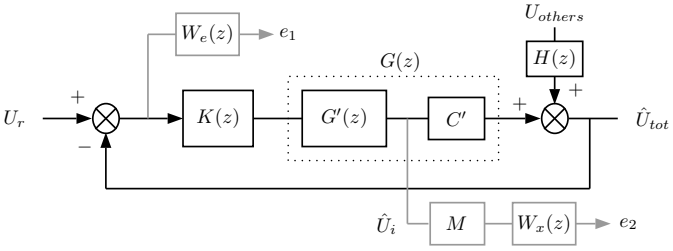

Fig. 4. H_∞ design bloc diagram

Scheduling Controller Design The bloc diagram of figure (4) is considered for the H_∞ design where $G'(z)$ is the model of the scheduler, the output of which is the vector of all task loads. To get the sum of all task loads, we use $C' = [1\ 1\ 1]$. The $H(z)$ transfer function represents the sensor dynamical behaviour which measures the load of the other tasks. It may be a first order filter. The template W_e specifies the performances on the load tracking error as follows:

$$W_e(s) = \frac{s/M_s + \omega_b}{s + \omega_s \epsilon} \tag{6}$$

with $M_s = 2$, $\omega_s = 10\, rad/s$, $\epsilon = 0.01$ to obtain a closed-loop settling time of $300\, ms$, a static error less than $1\,\%$ and a good robustness margin. Matrix M is defined as $M = [1\ -1\ -1]$.

The contribution of each of the compensation tasks to the controller performance w.r.t to its execution period has been evaluated via numerous simulations. However, due to the non-linear nature of the robot arm, only a very rough cost function could be identified : it appears that gravity compensation is the most important task therefore we have to allocate it more resources. The costs of Coriolis and inertia compensation are quite similar thus gravity compensation resources allocation is chosen to be twice of Coriolis or inertia ones.

The template W_x allows to specify the load allocation between the control tasks. With a large gain in W_x, it leads to:

$$U_{gravity} \approx U_{Coriolis} + U_{inertia},$$

i.e. we allocate more resources for the gravity compensation.

All templates are discretized with a sampling period of $30\ ms$. Finally discrete-time H_∞ synthesis algorithm produces a discrete-time scheduling controller of order 4.

Implementation of the Feedback Scheduler After preliminary simulations using TrueTime Cervin [2003] we have developed a feedback scheduler prototype running in real-time inside a "hardware-in-the-loop" simulator : a well calibrated model of the robot arm is numerically integrated in parallel with the execution of the controller, on top

of a real-time, preemptive and fixed priorities, operating system.

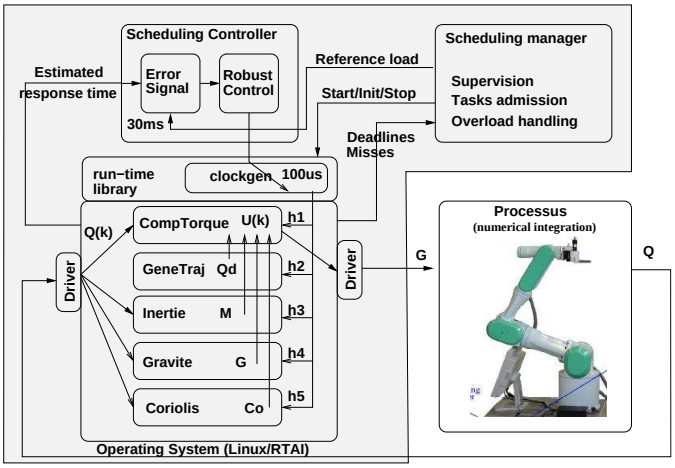

Fig. 5. Feed-back scheduling experiment

The process controller uses the so-called *Computing Torque Controller* which is split into several computing modules to implement a multi-rate controller as in Simon et al. [1998] (figure 5). The system is implemented using only the basic features of an off-the-shelf RTOS, which anyway must be instrumented with a task execution time operator [4]. In this application, the period of the feedback scheduler has been fixed to $30ms$ to be larger than the robot control tasks (which limits have been set here from $0.5ms$ to $30ms$).

In this experiment, due to the poor quality of the cost functions which were identified, the feedback scheduler directly controls the CPU usage rather than taking into account the state of the physical system as in an ideal case. In the experiment depicted in figure 6 the desired CPU usage is initially set to 60% of the maximum usage and then lowered to 40% after 1.5 sec. The upper plots show the tasks periods and CPU usage. Note that the processor also executes the robot arm numerical integration which induces a high and varying load, inducing some unpredictable overloads.

These first experimental results are encouraging : they show that such a feedback scheduling architecture can be quite easily designed and implemented on top of an off-the-shelf real-time operating system with fixed priority and preemption.

In this particular case the scheduling controller is a low order state feedback, which moreover is executed at a slow rate : hence its computing cost is very low (about $75\mu\ sec$ every $30ms$ on a 400 Mhz Pentium 2), i.e. less than 1% of the total control cost.

Indeed, compared with a fixed rate controller, the gain in control performance measured by the integrated tracking error is not impressive : this is due to the very rough modelling of the performance/control rate relationships of this non-linear system. The real improvement lies in the robustness of the system against transient overloads, and in the automatic setting of the tasks periods : the designer

[4] as in the several real-time variants of Linux we have used, i.e. RTAI(`www.rtai.org`) and Xenomai(`www.xenomai.org`)

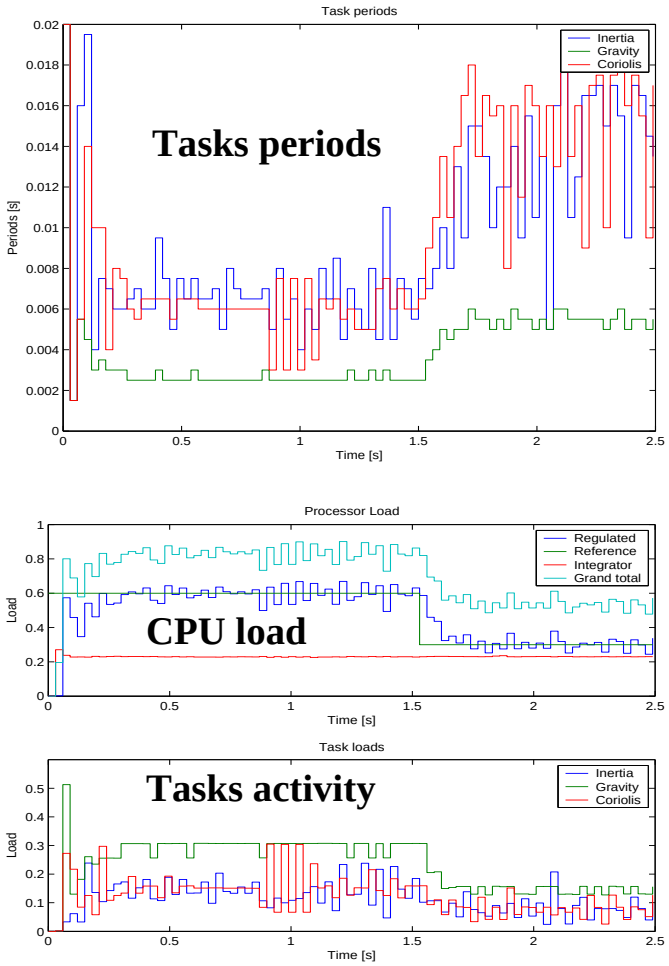

Fig. 6. Hardware in the loop simulation : periods and load

only needs to set reasonable initial values based on easily measured average execution times. As stated in Cervin [2005] it must be noticed that the recovery strategies used in the case of CPU overload can be selected in a set of predefined behaviours to improve the overall control performance. Here we used the *Skip* overrun processing, where the overrunning task finishes its current job but prevents its next expected schedule to be executed. Note that, thanks to the robustness of the closed-loop system w.r.t. jitter and occasional data loss, overruns must not be considered as fatal events as they would be in a system specified as "hard real-time".

From the sampled control point of view, it may be observed that abrupt and/or frequent period switches may lead to control instability, even if each periodic controller is stable for each constant sampling period, e.g. Schinkel et al. [2002]. Adding a low pass filtering template in the H_∞ scheduling controller here provides period variations smoother than the one provided by a simple period rescaling ; however this does not guarantee for stability and a wiser solution is looked for in the next section.

5. LPV SYNTHESIS OF A SAMPLING VARYING CONTROLLER

In this section we develop a varying sampling rate control algorithm based on modern gain scheduling design : it is

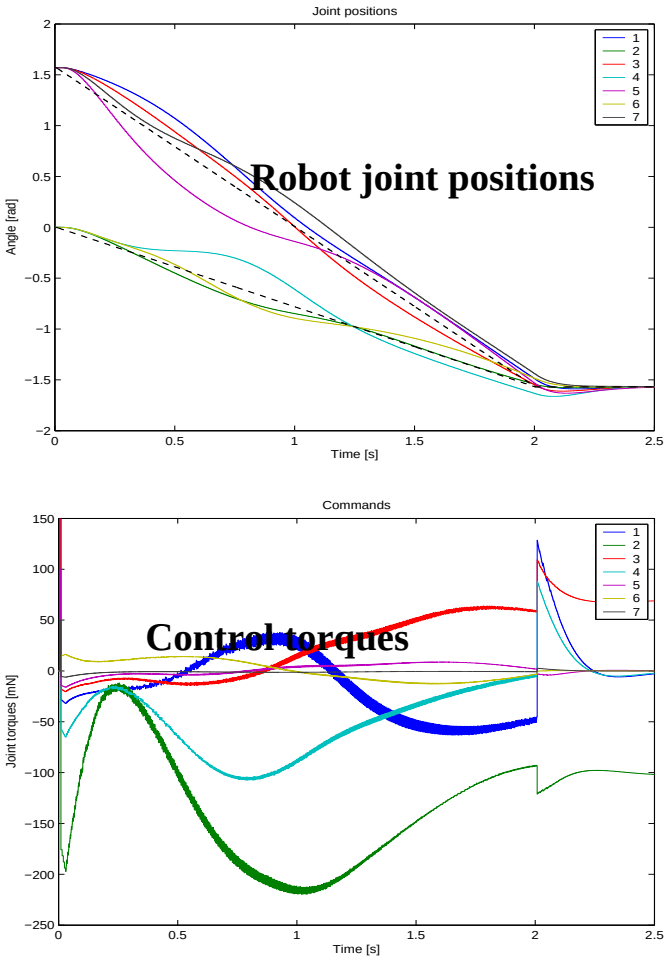

Fig. 7. Hardware in the loop simulation : analog results

able to guarantee the system's stability whatever are the instants and speed of variation of the control tasks periods. A specified performance level must be also preserved in the allowed period range. However this approach is up to now only designed for linear systems.

The main point is the problem formulation such that it can be solved following the LPV design of Apkarian et al. [1995]. Recall that this design ensures the stability and performance robustness of the closed loop parameter-varying system whatever are the variations of the parameters inside their predefined allowed range.

We propose a parametrised discretization of the continuous time plant and of the weighting functions, leading to a discrete-time sampling period dependent augmented plant. In particular the plant discretization approximates the matrix exponentials appearing in the discretized model by a Taylor series of order N. The original LPV design builds a discrete-time sampling period dependent controller through the convex combination of 2^N controllers, which may be conservative and complex to implement. In our particular case we exploit the dependency between the variables parameters, which are the successive powers of the sampling period $h, h^2, ..., h^N$, to reduce the number of controllers to be combined to $N+1$. This reduction of the polytopic set drastically decreases the conservatism of the original design and makes the solution easier to implement. We only provide here a summary of the approach which is described in details in Robert et al. [2007] and in Robert [2007].

5.1 A polytopic discrete-plant model

We consider a state space representation of continuous time plants as:

$$G : \begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \tag{7}$$

The exact discretization of this system with a zero order hold at the sampling period h can be computed (see Åström and Wittenmark [1997]) leading to the discrete-time LPV system (8)

$$G_d : \begin{cases} x_{k+1} = A_d(h)\, x_k + B_d(h)\, u_k \\ y_k = C_d(h)\, x_k + D_d(h)\, u_k \end{cases} \tag{8}$$

with h ranging in $[h_{min}; h_{max}]$. However computing A_d and B_d involve matrix exponentials of the original A and B matrices and thus are not affine on h.

To get a polytopic model and then apply an LPV design, we propose to approximate the exponential by a Taylor series of order N as:

$$e^{Mh} \approx \sum_{i=0}^{N} \frac{(Mh)^i}{i!}, \tag{9}$$

which leads, with $H = [h\ h^2\ \dots\ h^N]$, to

$$A_d(h) \approx I + \sum_{i=1}^{N} \frac{A^i}{i!}\, h^i := A_d(H) \tag{10}$$

$$B_d(h) \approx \sum_{i=1}^{N} \frac{A^{i-1}B}{i!}\, h^i := B_d(H) \tag{11}$$

Now the dependence on H is affine. To get a polytope $\mathcal{H}$ containing H, a solution is to choose $\mathcal{H}$ with the 2^N vertices ω_i corresponding to the vertices of the hypercube (13).

$$\mathcal{H} = \left\{ \sum_{i=1}^{2^N} \alpha_i(h)\omega_i : \alpha_i(h) \geq 0, \sum_{i=1}^{2^N} \alpha_i(h) = 1 \right\} \tag{12}$$

$$\{h, h^2, \dots, h^N\},\ h^i \in \{h^i_{min}, h^i_{max}\} \tag{13}$$

This leads to the plant polytopic model (14) where G_{d_i} are $G_d(H)$ evaluated at the vertices ω_i.

$$G_d(H) = \sum_{i=1}^{2^N} \alpha_i(h) G_{d_i} \quad \text{and} \quad H = \sum_{i=1}^{2^N} \alpha_i(h)\omega_i \tag{14}$$

As the gain-scheduled controller will be a convex combination of 2^N "vertex" controllers, the choice of the series order N gives a trade-off between the approximation accuracy and the controller complexity.

To decrease the volume and number of vertices of the matrices polytope we exploit the dependency between the successive powers of the parameter h. Recall that the vertices ω_i of $\mathcal{H}$ are defined by $h, h^2, \dots, h^N$ with $h^i \in \{h^i_{min}, h^i_{max}\}$. Indeed the representative point of the parameters set is constrained to be on a one dimensional curve, so that the polytope of interest can be reduced to the "lower $N+1$ vertices, as illustrated in figure 8 for the cases N = 2 and 3.

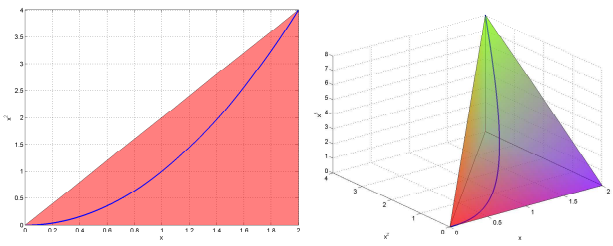

Fig. 8. Polytope reduction for N=2 and N=3

5.2 Performance specification

In the H_∞ framework, the general control configuration of figure 9 is considered, where W_i and W_o are weighting functions specifying closed-loop performances (see Skogestad and Postlethwaite [1996]). The objective is here to find a controller K such internal stability is achieved and $\|\tilde{z}\|_2 < \gamma\|\tilde{w}\|_2$, where γ represents the H_∞ attenuation level.

Classical control design assumes constant performance objectives and produces a controller with an unique sampling period. This sampling period is chosen according to the controller bandwidth, the noise sensibility and the availability of computation resources. When the sampling period varies, the usable controller bandwidth also varies and the closed-loop objectives should logically be adapted ; therefore we propose to adapt the bandwidth of the weighting functions. In this aim, W_i and W_o are split into two parts:

- a constant part with constant poles and zeros. This allows, for instance, to compensate for oscillations or flexible modes which are, by definition, independent of the sampling period. This part is merged with the plant before its discretization.
- the variable part contains poles and zeros whose pulsations are expressed as an affine function of the frequency $f = 1/h$. This makes possible to adapt the bandwidth of the weighting functions. These poles and zeros are here constrained to be *real* by the discretization step. Finally, opportune cancellations makes the *discretized* templates independent from the h, making easier further interconnections.

LPV/H_∞ control design The interconnection between the discrete-time polytopic model of the plant $\tilde{P}$ (now including the constant part of the weighting functions) and the variable weighting functions W_i and W_o leads to the discrete-time LPV augmented plant $P(H)$ is depicted in figure 9.

We aim to use here the H_∞ control design for linear parameter-varying systems detailed in Apkarian et al. [1995]. The method states that under some mild conditions, there exist a gain-scheduled controller :

$$\begin{cases} x_{K_{k+1}} &= A_K(H)x_{K_k} + B_K(H)y_k \\ u_k &= C_K(H)x_{K_k} + D_K(H)y_k \end{cases} \quad (15)$$

where $x_K \in \mathbb{R}^n$, ensuring over all parameter trajectories, for the closed-loop system:

- closed-loop quadratic stability

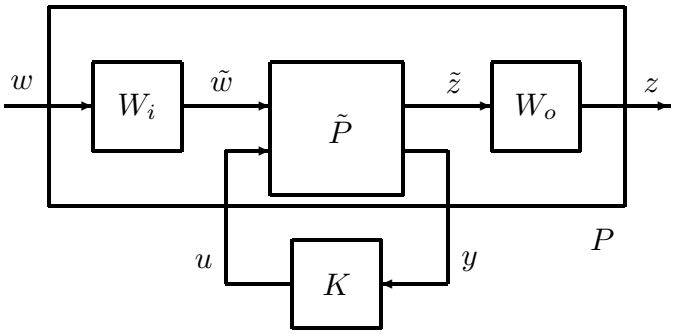

Fig. 9. Focused interconnection

- $\mathscr{L}_2$-induced norm of the operator mapping w into z bounded by γ, i.e. $\|z\|_2 < \gamma\|w\|_2$

$N+1$ controllers are reconstructed at each vertex of the parameter polytope (corresponding with the extrema values of the parameters). The gain-scheduled controller $K(H)$ is then the convex combination of these controllers

$$K(H) : \begin{pmatrix} A_K(H) & B_K(H) \\ C_K(H) & D_K(H) \end{pmatrix} = \sum_{i=1}^{r} \alpha_i(h) \begin{pmatrix} A_{K_i} & B_{K_i} \\ C_{K_i} & D_{K_i} \end{pmatrix} \quad (16)$$

$$\text{with } \alpha_i(h) \text{ such that } H = \sum_{i=1}^{r} \alpha_i(h)\omega_i \quad (17)$$

Note that on-line scheduling of the controller needs the computation of $\alpha_i(h)$ knowing h. Considering a Taylor's expansion around h_0 with

$$\delta_{min} = h_{min} - h_0 \text{ and } \delta_{max} = h_{max} - h_0$$

and the case of the reduced polytope, explicit solutions are easily recursively computed:

$$\begin{cases} \alpha_1 = \dfrac{\delta_{max} - \delta}{\delta_{max} - \delta_{min}} \\ \alpha_n = \dfrac{\delta_{max}^n - \delta^n}{\delta_{max}^n - \delta_{min}^n} - \displaystyle\sum_1^{n-1} \alpha_i \quad , \quad n = [2, ..., N] \\ \alpha_{N+1} = 1 - \displaystyle\sum_1^{N} \alpha_i \end{cases}$$

This leads, for the case $N = 2$ and $\delta_{min} = 0$ of the next section to the simple explicit solutions:

$$\alpha_1 = \frac{\delta_{max} - \delta}{\delta_{max}}, \alpha_2 = \frac{\delta_{max}^2 - \delta^2}{\delta_{max}^2} - \alpha_1, \alpha_3 = 1 - (\alpha_1 + \alpha_2)$$

5.3 Experimental assessment

The latter approach has been experimentally assessed using a "T" inverted pendulum of Educational Control Products [5], available at Gipsa-lab, in the NeCS (Network Controlled Systems [6]) project. These experiments will emphasise the pertinence of the proposed design method.

The pendulum depicted in figure 10 is composed of two rods. A vertical one which rotates around the pivot axle,

[5] www.ecpsystems.com/controls_pendulum.htm
[6] http://necs.inrialpes.fr

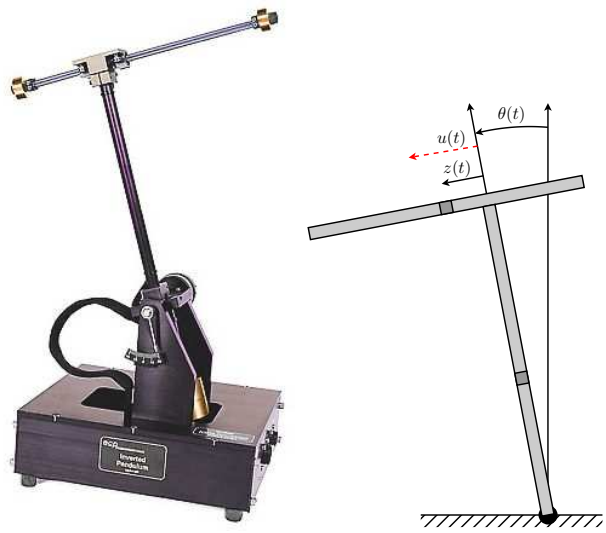

Fig. 10. The T pendulum under experiment

and an horizontal sliding balance one. Two optional masses allow to modify the plan dynamical behaviour.

The control actuator (DC motor) delivers a force u to the horizontal sliding rod, through a drive gear-rack. The θ angle, positive in the trigonometric sense, is measured by the rod angle sensor. The position z of the horizontal rod is measured by a sensor located at the motor axle. The DC motor is torque controlled using a local current feedback loop (assumed to be a simple gain due to the high dynamics). The dynamical behaviour of the sensors is also neglected.

As such a T pendulum system is difficult to be controlled, our main objective is here to get a closed-loop stable system, to emphasise the practical feasibility of the proposed methodology for real-time control. The sampling period is assumed to be in the interval $[1, 3]$ ms.

The chosen performance objectives are represented in figure 11, where the tracking error and the control input are weighted (as usual in the H_∞ methodology).

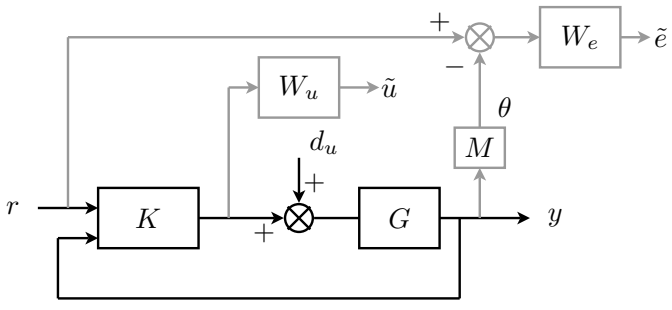

Fig. 11. Control configuration

This corresponds to the mixed sensitivity problem given in (18).

$$\left\| \begin{matrix} W_e(I - MS_yGK_1) & W_eMS_yG \\ W_uS_uK_1 & W_uT_u \end{matrix} \right\|_\infty \leq \gamma \qquad (18)$$

with

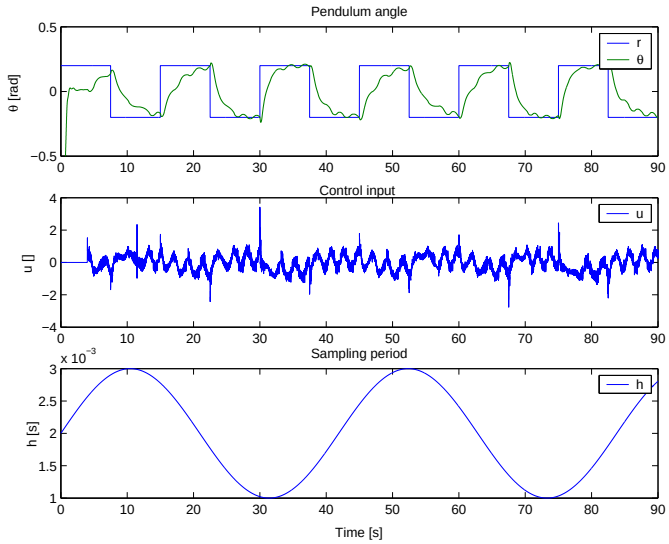

Fig. 12. Experimental motion of the T pendulum under a sinusoidal sampling period

$$\begin{aligned} K &= [K_1 \ K_2] & M &= [0 \ 0 \ 1 \ 0] \\ S_u &= (I - K_2G)^{-1} & S_y &= (I - GK_2)^{-1} \\ T_u &= -K_2G(I - K_2G)^{-1} \end{aligned} \qquad (19)$$

The performance objectives are represented by weighting functions and may be given by the usual transfer functions Skogestad and Postlethwaite [1996]:

$$W_e(p, f) = \frac{p\, M_S + \omega_S(f)}{p + \omega_S\, \epsilon_S}, \omega_S(f) = h_{min}\, \omega_{S_{max}}\, f \qquad (20)$$

$$W_u(p, f) = \frac{1}{M_U} \qquad (21)$$

where $f = 1/h$, $\omega_{S_{max}} = 1{,}5$ rad/s, $M_S = 2$, $\epsilon_S = 0.01$ and $M_U = 5$.

For implementation reasons (simplicity and computational complexity) we have chosen the case of the reduced polytope using a Taylor expansion of order 2, leading to a reduced polytope with 3 vertices.

Experiments results The plant is controlled through Matlab/Simulink using the Real-time Workshop and xPC Target. Two cases are presented. First in figure 12 the sampling period variation is continuous and follows a sinusoidal signal of frequency $0.15 rad/s$. Then in figure 13 some step changes of the sampling period are done.

As expected from the sampling dependent performance objectives, the settling time is minimal when the sampling period is maximal, and conversely. There are no abrupt changes in the control signal, even when the sampling period suddenly varies from 1 to 3 ms as in figure 13.

Therefore this design method appears to be effective to preserve the plant's stability and performance objectives during arbitrarily fast control periods variations, that can further used to cope with varying computing resources availability.

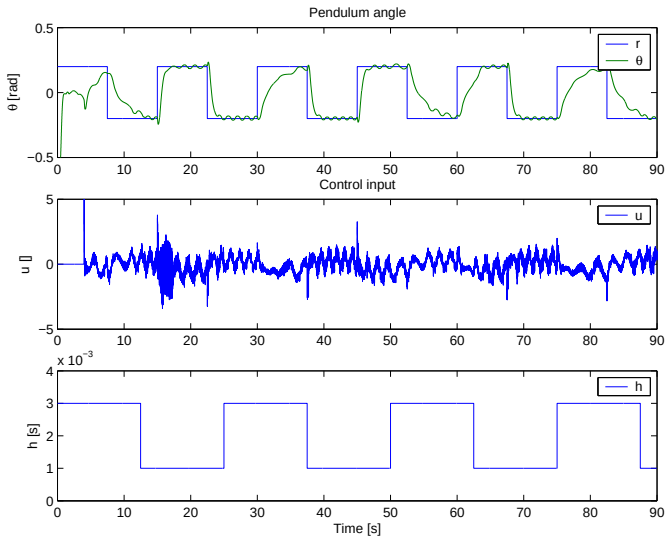

Fig. 13. Experimental motion of the T pendulum under a square sampling period

6. SUMMARY AND FURTHER DIRECTIONS

In this paper, some methodologies for robust control and scheduling co-design were proposed. While robust control usually deals with modelling errors of the controlled plant, a digital closed-loop controller is also subjected to timing disturbances coming from its implementation using a real-time operating system. These disturbances are difficult to be accurately predicted, and studying the impact of timing deviations in feedback loops is still a largely unexplored domain. Hence a natural idea consists in using robust control theory to design controllers to be weakly timing sensitive. Besides process control this idea can be used also to design a feedback scheduling loop to implement robust on-line adaption of the scheduling parameters according to estimates of the computing activity. Some partial solutions for computing resource control and allocation has been presented in the paper. Among others, the two following research directions deserve to be further investigated :

6.1 Adaptive scheduling and robust control

Variable sampling rate appears to be a decisive actuator in scheduling and CPU load control. Although it is quite conservative, the LPV based design developed in section 5 guarantees plant stability and performance level, whatever is the speed of variation of the control period inside its predefined range. Hence the control tasks periods of such controllers can be adapted on-line by an external loop (the feedback scheduler) on the basis of resource allocation and global quality of service (QoS), with no further care about the process control stability. Hence a quite simple scheduling control architecture, e.g. like a simple rescaling as proposed in Cervin [2003], or an elastic scheduler as in Buttazzo and Abeni [2000].

Indeed, besides the flexibility and robustness provided by an adaptive scheduling, a full benefit would come by taking into account directly the controlled process state in the scheduling loop. It has been shown in Eker et al. [2000] that even for simple case the full theoretic solution based on optimal control was too complex to be implemented in real-time. However it is possible to sketch effective solutions suited for some case studies as depicted in figure 14 taken from Robert [2007].

Conversely with the robot controller in section 4.4 the load allocation ratio between the control components is no longer constant and defined at design time. It is made dependent on the measure of the quality of control (QoC) to give advantage to the controller with higher control error. The approach relies on a modified elastic scheduler algorithm, where the "stiffness" of every control task depends on the control (through the M_i component in figure 14 which can be a simple gain). The approach is still simple to implement and, even if only tested in simulation up to now, has shown significant performance improvements compared with more simple (i.e. control quality unaware) resource allocation.

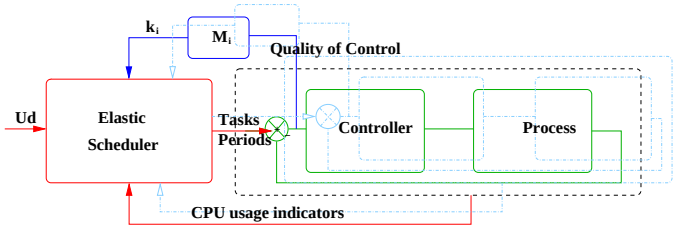

Fig. 14. Integrated control/scheduling loops

However, the dynamic of the scheduling loop now includes the scheduling dependent dynamics of the process itself. Ensuring the stability of this integrated control/scheduling loop requires an adequate modelling of the relationships between the control quality and the scheduling parameters, which is still to be done in a general case.

6.2 Accelerable control tasks

A common assumption about the sampling rate of control tasks is that faster is the computation and better is the result, i.e. that control tasks are always accelerable. However recent investigations Ben Gaid et al. [to appear] revisited this assumption and indicated new directions.

An accelerable control task has the property that more executions are performed, better is the control performance. When used in conjunction with weakly-hard real-time scheduling design, an accelerable control task allows taking advantage of the extra computational resources that may be allocated to it, and to improve the control performance with respect to worst case design methods. In practice, however, control laws designed using standard control design methods (assuming a periodic sampling and actuation) are not necessarily accelerable. Case studies have shown that, when a control law is executed more often than allowed by the worst case execution pattern (with no gains adaption), the performance improvement may be state dependent, and that performance degradation can be observed.

Conditions for the design of accelerable tasks has been established, based on Bellman optimality principle, in the framework of a (m,k)-firm scheduling policy. Is is assumed that an optimal control law exists in the case of the worst case execution sequence, i.e. when only the *mandatory* instances of the control task are executed to completion.

It is shown that, in the chosen framework, it is possible to systematically design accelerable control laws, which control performance increases when more instances of the control task can be executed during the free time slots of the system. The theory is quite general and does not require the process linearity.

REFERENCES

P. Apkarian, P. Gahinet, and G. Becker. Self-scheduled H_∞ control of linear parameter-varying systems: A design example. *Automatica*, 31(9):1251–1262, 1995. ISSN 0005-1098.

K.J. Åström and B. Wittenmark. *Computer-Controlled Systems*. Prentice Hall, 3rd edition, 1997.

M-M. Ben Gaid, A. Çela, Y. Hamam, and C. Ionete. Optimal scheduling of control tasks with state feedback resource allocation. In *Proceedings of the 2006 American Control Conference*, Minneapolis, USA, June 2006.

M-M. Ben Gaid, D. Simon, and O. Sename. A design methodology for weakly-hard real-time control. Technical report, INRIA, to appear.

G. Bernat, A. Burns, and A. Llamosí. Weakly hard real-time systems. *IEEE Transactions on Computers*, 50(4): 308–321, 2001.

G. Buttazzo and L. Abeni. Adaptive rate control through elastic scheduling. In *39th Conference on Decision and Control*, Sydney, Australia, 2000.

G. Buttazzo and A. Cervin. Comparative assessment and evaluation of jitter control methods. In *Proc. 15th International Conference on Real-Time and Network Systems*, Nancy, France, March 2007.

A. Cervin. *Integrated Control and Real-Time Scheduling*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Sweden, April 2003.

A. Cervin and J. Eker. Feedback scheduling of contol tasks. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, December 2000.

A. Cervin, J. Eker, B. Bernhardsson, and K. Årzén. Feedback-feedforward scheduling of control tasks. *Real-Time Systems*, 23(1–2):25–53, July 2002.

Anton Cervin. Analysis of overrun strategies in periodic control tasks. In *Proc. 16th IFAC World Congress*, Prague, Czech Republic, July 2005.

J. Eker, P. Hagander, and K-E. Årzén. A feedback scheduler for real-time controller tasks. *Control Engineering Practice*, 8(12):pp 1369–1378, 2000.

N. Jia, Y.-Q. Song, and F. Simonot-Lion. Graceful degradation of the quality of control through data drop policy. In *Proceedings of the European Control Conference*, Kos, Greece, July 2007.

C. Lu, J.-A. Stankovic, G. Tao, and S.-H. Son. Feedback control real-time scheduling: Framework, modeling, and algorithms. *Real Time Systems*, 23(1):85–126, 2002. ISSN 0922-6443.

D. Robert, O. Sename, and D. Simon. A reduced polytopic LPV synthesis for a sampling varying controller : experimentation with a T inverted pendulum. In *European Control Conference ECC'07*, Kos, Greece, 2007.

David Robert. *Contribution à l'interaction commande/ordonnancement*. PhD thesis, I.N.P. Grenoble, janvier 2007.

M. Ryu, S. Hong, and M. Saksena. Streamlining real-time controller design: from performance specifications to end-to-end timing constraints. In *IEEE Real Time Systems Symposium*, 1997.

K. Sandström and C. Norström. Managing complex temporal requirements in real-time control systems. In *9th IEEE Int. Conf. and Workshop on the Engineering of Computer-Based Systems (ECBS'02)*, Lund, Sweden, 2002.

M. Schinkel, W-H Chen, and A. Rantzer. Optimal control for systems with varying sampling rate. In *Proceedings of American Control Conference*, Anchorage, May 2002.

D. Simon, E. Castillo, and P. Freedman. Design and analysis of synchronization for real-time closed-loop control in robotics. *IEEE Trans. on Control Systems Technology*, 6(4):445–461, july 1998.

D. Simon, O. Sename, D. Robert, and O. Testa. Real-time and delay-dependent control co-design through feedback scheduling. In *CERTS'03 Workshop on Co-design in Embedded Real-time Systems*, Porto, july 2003. ECRTS.

D. Simon, D. Robert, and O. Sename. Robust control / scheduling co-design: application to robot control. In *Proceedings of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Francisco, USA, March 2005.

S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control*. John Wiley and Sons, 1996.

M. Törngren. Fundamentals of implementing real-time control applications in distributed computer systems. *Real Time Systems*, 14(3):219–250, 1998.

B. Wittenmark. A sample-induced delays in synchronous multirate systems. In *European Control Conference*, pages 3276–3281, Porto, Portugal, 2001.

K. Zhou, J. C. Doyle, and K. Glover. *Robust and optimal control*. Prentice-Hall Inc., 1996.